Tiedon implementointi Microsoft Dynamics AX 2009 -järjestelmään

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, 25.8.2010.

Harri Malkamäki

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Työn nimi	Tiedon implementointi Microsoft Dynamics AX 2009 -järjestelmään
Tekijä	Harri Malkamäki
Ohjaava opettaja	Lasse Seppänen
Hyväksytty	25.8.2010
Hyväksyjä	

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Hämeenlinna
Tietojenkäsittelyn koulutusohjelma

Tekijä Harri Malkamäki **Vuosi** 2010

Työn nimi Tiedon implementointi Microsoft Dynamics AX 2009 -järjestelmään

TIIVISTELMÄ

Opinnäytetyö perustuu GS-Hydro Oy:ltä saatuun toimeksiantoon. GS-Hydro Oy:ssä oli käynnissä toiminnanohjausjärjestelmän vaihdos, jossa piti toteuttaa tietojen implementointi. Tietoa piti siis siirtää vanhasta järjestelmästä sekä muista sähköisistä lähteistä saaduista tiedoista uuteen järjestelmään. Alustavaa tietoa opinnäytetyön tekijällä oli relaatiotietokannoista sekä SQL-kielestä, mutta projektin aikana oli opeteltava paljon uusia asioita, kuten kokonaan uusi ohjelmointiympäristö ja -kieli.

Toimeksiannon käytännön osuudessa suunniteltiin ja toteutettiin tiedon implementointi Microsoft Dynamics AX -järjestelmään. Järjestelmään siirrettiin projektin aikana useita tuhansia rivejä tietoa. Käytännön osuus saatiin toteutettua suunnitellussa aikataulussa. Opinnäytetyön sisältö on dokumentaatiota käytännön osuuteen vaadittavista tiedoista sekä ohjeistus tuotetietojen ja tilaajien sekä toimittajien implementoinnista järjestelmään.

Teoriaosuudessa esitellään yleisellä tasolla ERP- eli toiminnanohjausjärjestelmät. Lisäksi kerrotaan relaatiotietokannoista ja SQL-kielestä, koska ne liittyvät olennaisesti toiminnanohjausjärjestelmien toimintaan. Tämän jälkeen käydään läpi Dynamics AX:n käyttöliittymä. Tämä käyttöliittymä- tarkastelu tehdään pääasiassa ohjelmointi- ja tiedonsiirtonäkökulmasta.

Opinnäytetyön loppuosuudessa esitellään ensin tuotetiedon siirtäminen järjestelmään, mikä tarkoittaa perustietoja sekä reittejä ja rakenteita. Lopuksi työssä käydään läpi tilaaja- ja toimittajatietojen siirto. Yhteenvedossa kerrotaan vielä projektin onnistumisesta yleensä sekä käytännönosuuden aikana kohdatuista ongelmista.

Avainsanat Toiminnanohjausjärjestelmä, tietokanta, tiedonsiirto, ohjelmointi

Sivut 27 s. + liitteet 2 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Hämeenlinna
Degree Programme in Business Information Technology

Author Harri Malkamäki **Year** 2010

Subject of Bachelor's thesis Implementing data to Microsoft Dynamics AX 2009

ABSTRACT

This thesis is based on an assignment from GS-Hydro. The company was going through an enterprise resource planning system change in which a data implementation process had to be carried out. Data had to be transferred from the old system and from other electronic sources to a new system. I was already familiar with relational databases and SQL-language, but during the project I had to learn many new things like a completely new programming environment and a new programming language.

In the practical part of the thesis the data implementation to Microsoft Dynamics AX system was planned and executed. Several thousands of rows of data were transferred to the system. The practical part was accomplished in schedule. This thesis consists of the documentation about the information needed for the practical part and instructions for the implementation of product data and vendor and customer data.

In the theoretical part of the thesis enterprise resource planning systems are introduced on a general level. Relational databases and SQL-language are also mentioned because they are relevant to enterprise resource planning systems. After this the Dynamics AX user interface is introduced. This interface analysis is made mainly in the programming and data transfer perspective.

In the last part of the thesis the implementation of product data to the system is introduced first, meaning basic data, routes and bills of materials. Lastly the implementation of customers and vendors is explained. In the summary I evaluate how well the project succeeded overall and about the problems met during the practical part of the thesis.

Keywords enterprise resource planning system, database, data transfer, programming

Pages 27 p. + appendices 2 p.

SISÄLLYS

Sanasto

Arkkitehtuuri	Järjestelmien yleinen rakenne ja rakenneperiaate.
DMX	Data Mining Extensions; Microsoftin kehittämä tietokannan kyselykieli.
Implementoida	Toteuttaa jokin tekninen periaateratkaisu käytännössä laitteella tai järjestelmällä.
IDE	Integrated Development Environment; ohjelmointiympäristö
Konversiokoodi	Ohjelmakoodi, joka ajetaan tiedonsiirron yhteydessä.
Moduuli	Näkymien (form), raporttien ja muiden AX:n toiminnallisuuksien joukko.
Replikointi	Tiedon monistaminen.
Tietolouhinta	Prosessi, jossa tiedosta etsitään yhteneväisyyksiä.

1 JOHDANTO

ERP- (Enterprise Resource Planning) eli toiminnanohjausjärjestelmä on yrityksen toiminnan kannalta tärkeä, ellei tärkein, työkalu yrityksestä riippumatta. Yrityksen toiminta koostuu aina useammasta osa-alueesta, kuten laskutuksesta, tuotannosta, jakelusta, varastonhallinnasta ja kirjanpidosta. Yrityksen koon kasvaessa myös nämä osa-alueet kasvavat ja vaativat enemmän resursseja, joko materiaalia tai henkilötyövoimaa. Tällöin yrityksessä kulkevan tiedon määrä kasvaa kovalla vauhdilla. Jotta yrityksen toiminta ei sirpaloituisi, tarvitaan yksi ydinjärjestelmä, johon kootaan tietoa ja jolla ohjataan yrityksen toimintaa joko lokaalissa tai globaalissa mittakaavassa. Toiminnanohjausjärjestelmien tarkoituksena on siis ohjata ja integroida yritysten eri osa-alueita yhteen eli tehostaa yrityksen toimintaa.

Erilaisia ERP-järjestelmiä on käyttötarkoituksesta riippuen useita, esim. Microsoft tarjoaa pienille yrityksille Microsoft Dynamics NAV -järjestelmää ja isoille yrityksille Microsoft Dynamics AX -järjestelmää. Yritykset voivat myös halutessaan toteuttaa ERP-järjestelmänsä täysin räätälöityinä ratkaisuina alusta alkaen. Toiminnanohjausjärjestelmän käyttöönotto on yritykselle kuitenkin aina monimutkainen sekä rahaa ja aikaa vaativa operaatio. Järjestelmän kehitykseen ennen käyttöönottoa kannattaa panostaa, jotta siitä saadaan mahdollisimman nopeasti maksimaalinen hyöty. Jotta järjestelmä saataisiin toimimaan halutulla tavalla ja muokattua yrityksen toimintaan sopivaksi, vaaditaan yleensä aina ulkopuolista asiantuntemusta järjestelmään liittyen. Ulkopuolinen asiantuntemus on erityisen tärkeää isompien yritysten kohdalla, sillä niissä toiminnanohjausjärjestelmät ovat laajoja ja integroituja kokonaisuuksia.

GS-Hydro Oy on suomalainen, pääasiassa putkistojärjestelmiä teollisuuden, marine- ja offshoretoiminnan sovelluksiin valmistava yritys. Yrityksen päätoimipiste on Hämeenlinnassa ja sen toiminta on maailmanlaajuista. Toimipisteitä GS-Hydrolla on 17 maassa. Yrityksessä alettiin noin vuosi sitten siirtyä uuteen ERP-järjestelmään vanhan ollessa globaalin toiminnan kannalta jo vanhentunut. Järjestelmäksi valittiin Microsoft Dynamics AX 2009, joka tarjoaa hyvät mahdollisuudet usean yrityksen hallintaan saman ydinjärjestelmän kautta. AX:ään on myös helppo lisätä tarvittaessa erilaisia moduuleita ja se on miltei täysin muokattavissa yrityksen tarpeita vastaavaksi. Tehtävänäni projektissa on tutkia ja toteuttaa tiedonsiirto tämänhetkisestä järjestelmästä uuteen järjestelmään. Käytännössä tämä tarkoittaa tuotetiedon sekä tilaaja- ja toimittajatietojen siirtoa. ERP-järjestelmissä tieto on yleensä aina tallennettuna tietokantaan, joka sisältää mahdollisesti useita satoja tietokannan tauluja. Tutkimuksen lähtökohtana on tutustua tapaan, jolla tieto on varastoitu järjestelmissä.

2 ERP-JÄRJESTELMÄT

ERP-järjestelmät ovat monimutkaisia kokonaisuuksia, joten niiden esitteleminen kovin syvällisesti ei ole tässä yhteydessä järkevää. Seuraavassa luvussa esitellään siis järjestelmät yleisellä tasolla. Myös Microsoft Dynamics AX 2009 -järjestelmä esitellään tässä luvussa yleisellä tasolla. Myöhemmin paneudutaan syvällisemmin tiedon implementointiin liittyviin järjestelmäosioihin.

2.1 ERP-järjestelmät yleisesti

Seuraavien kappaleiden lähteenä on käytetty Monkin ja Wagnerin (2009) kirjaa. Heidän mukaansa ERP- eli toiminnanohjausjärjestelmät ovat yritysten ydinsovelluksia, joilla koordinoidaan tiedon kulkua yrityksen liiketoiminnan jokaisella osa-alueella. Toiminnanohjausjärjestelmät helpottavat yrityksen liiketoimintaprosesseja yhdistämällä eri osa-alueiden tiedot yhteen tietokantaan ja jaettuihin raportointityökaluihin. Ne ovat siis monimutkaisia kokonaisuuksia, joiden ymmärtämiseksi täytyy ensin tietää perusasioita liiketoiminnasta.

Suurimmalla osalla yrityksistä on neljä yritystoiminnan osa-aluetta, jotka ovat markkinointi ja myynti, tuotannonohjaus, kirjanpito ja talous sekä henkilöstön hallinta. Näistä jokainen osa-alue voidaan vielä jakaa pienempiin liiketoiminnan osa-alueisiin. Jokainen osa-alue on toisistaan riippuvainen vaatien tietoa toiselta alueelta toimiakseen tehokkaasti. Mitä paremmin yritys pystyy integroimaan nämä osa-alueet toisiinsa, sitä paremmin se menestyy nykypäivän kilpailutilanteessa. Tarkasteltaessa tarkemmin jokaista osa-aluetta ymmärtää paremmin, miten ne ovat sitoutuneet toisiinsa.

Markkinoinnin ja myynnin tehtävinä ovat tuotekehitys, hinnoittelu, tuotteiden myynti ja mainostaminen asiakkaille sekä tilausten vastaanotto. Tämä osasto helpottaa myös ennusteiden tekemistä yritykselle. Tuotekehitys on luonnollisesti yritykselle tärkeää. Tuotekehitystä voidaan tehdä tutkimalla myyntihistoriaa ja etsimällä siitä trendejä. Tuotteen myynnin kannattavuus riippuu myös siitä, mitä sen tuottaminen maksaa. Kannattavuus voidaan selvittää tutkimalla tuotannonohjauksen tietoa tuotantokustannuksista ja materiaalien hinnoista.

Tuotannonohjauksen toimintoina ovat tuotteiden valmistus sekä raaka-aineiden hankinta. Tuotanto suunnitellaan niin, että tuotteita on mahdollisimman paljon saatavilla mutta ei kuitenkaan niin paljon, että niitä pitäisi hävittää vanhenemisen takia. Tässä suunnittelussa hyödynnetään paljon myynnin ja markkinoinnin tuottamia myyntiennusteita. Tuotannon suunnittelun ennusteet tarkentuvat huomattavasti, kun tiedetään, paljonko myyntiä on mahdollisesti tulevaisuudessa. Näin ollen markkinoinnin ja myynnin osa-alueella on paljon merkitystä tuotannonohjauksessa.

Kirjanpito ja talous vastaavat transaktiotietojen keräämisestä, raaka-aineiden ostoista, palkkojen maksusta sekä asiakkailta saatujen tulojen kirjaamisesta. Talousosasto siis kerää yhteen raa'an tiedon selkeiksi yhteenvedoiksi, joista selviää liiketoiminnan kannattavuus. Näitä tietoja tarvitaan sekä markkinoinnin että tuotannon osa-alueilla.

Henkilöstön hallinnan tehtävänä on yrityksen henkilöstön palkkaaminen, koulutus, evaluointi ja hyvittäminen eli palkan maksu. Henkilöstön hallinta tarkastelee jatkuvasti yrityksen työvoiman tarvetta ja työtilannetta. Näin ollen sen toiminta on tarpeellista jokaiselle muulle yritystoiminnan osa-alueelle. Toisaalta se tarvitsee myös tietoa sekä myynnistä että tuotannosta, jotta tarvittava työvoiman määrä voidaan arvioida.

Yrityksen jokainen osa-alue on siis tavalla tai toisella riippuvainen toisesta ja tiedon välitys eri osa-alueiden välillä on kriittistä yrityksen toiminnan kannalta. Toiminnanohjausjärjestelmä integroi nämä järjestelmät toisiinsa ja tarjoaa tarvittavan tiedon jakamisen osien välillä. (Monk & Wagner 2009, 2-6.)

Toiminnanohjausjärjestelmien käyttöönotto vaatii paljon asiantuntemusta. Näin ollen käyttöönotossa tarvitaan usein ulkopuolista konsultointia. Ensimmäiset ERP-järjestelmät on kehitetty vasta viime vuosikymmeninä niiden vaatiman teknologian takia. Suuret järjestelmät vaativat laajan tietokannan toimiakseen, mikä taas vaatii paljon levytilaa sekä laskentatehoa tietokoneelta. Nykyiset toiminnanohjausjärjestelmät vaativat käytännössä aina toimiakseen tietokantapalvelimen. Vaadittu tietokantapalvelimen tarjoaja riippuu ERP-palvelun tarjoajasta, mutta useimmat järjestelmät yhteensopivat monen eri tietokantapalvelimen kanssa.

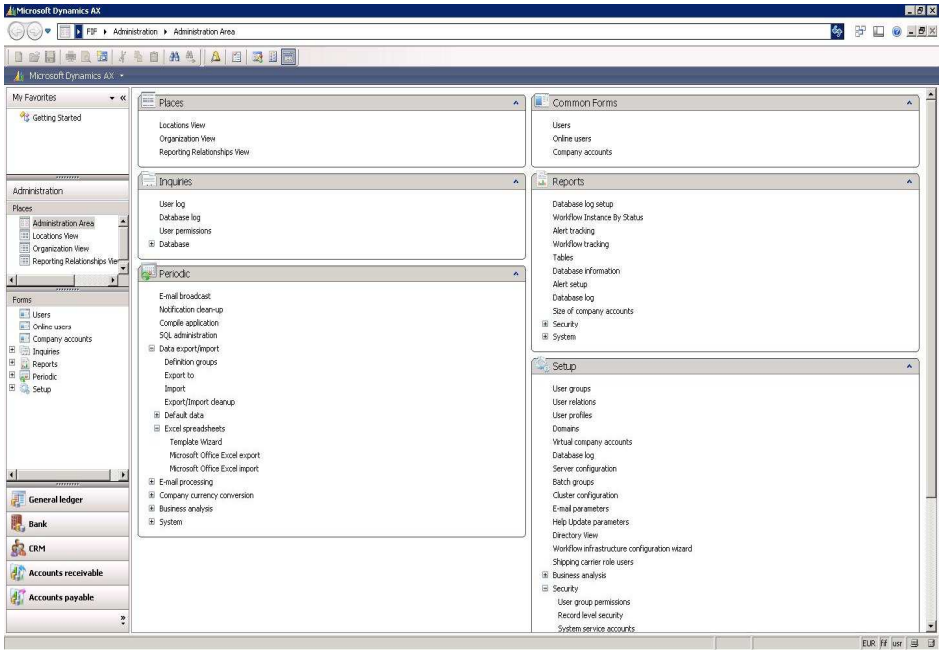
Useimmat nykyisistä toiminnanohjausjärjestelmistä rakentuvat erilaisista moduuleista. Moduuleiden tarkoituksena on jakaa järjestelmät loogisesti eri osa-alueisiin. Niiden hyötynä on myös se, että toiminnanohjausjärjestelmää valitseva yritys voi räätälöidä sovelluksen omiin toimintoihinsa sopivaksi suhteellisen helposti tiputtamalla pois tarpeettomia moduuleita. Näin voidaan tarvittaessa säästää hankintakustannuksissa. Moduulien sisältö sekä se, mitä moduuleita tarjotaan, riippuu käytettävästä toiminnanohjausjärjestelmästä. Useissa järjestelmissä myös kolmannet osapuolet voivat luoda omia räätälöityjä moduuleitaan eri yrityksille.

Monkin ja Wagnerin (2009) mukaan tällä hetkellä suurin ERP-järjestelmän tarjoaja on saksalainen SAP. SAP oli myös ensimmäinen yritys, joka kehitti ohjelmistoja ERP-järjestelmiin. Erilaisia toiminnanohjausjärjestelmiä ja niiden tarjoajia on nykyään paljon. Järjestelmien käyttö ei rajoitu enää vain isoihin yrityksiin, sillä eri tarjoajat ovat räätälöineet ERP-paketteja myös pienille ja keskisuurille yrityksille.

2.2 Microsoft Dynamics AX 2009

AX, entiseltä nimeltään Axapta, on Microsoftin Dynamics tuoteperheeseen kuuluva ohjelmistotuote. Se on keskisuurille ja suurille yrityksille tarkoitettu toiminnanohjausjärjestelmä, jonka tällä hetkellä uusin versio ilmestyi vuonna 2009. AX:n kehityksen aloitti vuonna 1983 Damgaard-niminen tanskalainen yritys, joka vuonna 2000 yhdistyi Navision Softwaren kanssa. Microsoftin hallussa tuote on ollut vuodesta 2002, kun Microsoft osti Navision A/S:n. Käyttöliittymältään AX muistuttaa hyvin paljon Microsoftin muita tuotteita, kuten Microsoft Office -tuoteperheen ohjelmistoja. Kehitysympäristö taas muistuttaa paljon esimerkiksi Visual Basicin käyttöliittymää.

Dynamics AX:n arkkitehtuuri koostuu useampaa käyttäjää varten skaalautuvista käyttäjä- ja palvelinkomponenteista. Se on myös yhteentoimiva useimpien muiden Microsoftin tuotteiden, kuten Microsoft Excelin kanssa. Arkkitehtuuri sisältää sovellusmallin ja X++-ohjelmointikielen sekä Morphx-sovelluskehitysympäristön. Arkkitehtuuri sisältää myös sovelluskehyksen. AX:n arkkitehtuuri on suunniteltu ohjelmistokehittäjien kannalta niin, että sillä on mahdollisimman helppo rakentaa ja kehittää omia ohjelmia vähentämällä itse ohjelmakoodin kirjoittamisen tarvetta. AX siis huolehtii automaattisesti esimerkiksi käyttöliittymän, käyttäjä- ja palvelinrajapinnan sekä tietokantayhteyden toteuttamisesta niin, että sovelluskehittäjä voi keskittyä liiketoiminnan kannalta oleelliseen kehitykseen. (The Microsoft Dynamics AX Team 2009.)



Kuva 1. *Microsoft Dynamics AX.*

AX on jaettu eri osa-alueisiin toiminnallisuuksien mukaan. Näitä erilaisia osa-alueita kutsutaan moduuleiksi. AX koostuu 19 moduulista, jotka on rakennettu toisistaan riippumattomiksi osa-alueiksi, eli ne ovat käytännössä

vapaasti liitettävissä tai poistettavissa riippuen yrityksen tarpeesta. AX:n perusmoduuleita ovat taloushallinto, projektihallinto, asiakkuudenhallinta, henkilöstöhallinta, toimitusketjun hallinta, liiketoiminnan analysointi, tuotannonohjaus ja sähköinen kauppa.

3 TIETOKANNAT

Tietokanta on käytännössä tietovarasto, jota järjestelmä käyttää tiedon varastointiin jonkin säännön mukaan niin, että tieto pysyy järjestyksessä. Tietokantojen koko riippuu täysin järjestelmän koosta. Kanta voi koostua esimerkiksi vain yhdestä tekstitiedostosta tai sitten useista sadoista tai tuhansista tietokantatauluista. Toiminnanohjausjärjestelmien tietokannat ovat aina isoja ja monimutkaisia kokonaisuuksia. Nykyiset tietokannat toimivat usein erillisellä tietokantapalvelimella, joita hallinnoidaan eri palveluntarjoajien tarjoamilla tietokantapalvelinohjelmistoilla. Tässä luvussa esitellään Microsoftin SQL Server –palvelin, koska AX on optimoitu toimimaan tällä tietokantapalvelimella.

3.1 Relaatiotietokannat

Nykyisistä käytössä olevista tietokantamalleista yleisin on relaatiotietokantamalli. Relaatiotietokannat koostuvat juuritasosta ja sen alitasoista. Näin tietokantaan syntyy hierarkkinen rakenne. Sen mallin ja käsittelymääritelmän kehitti jo 1970-luvulla E. F. Codd, joka työskenteli tuolloin IBM:n palveluksessa. Määritelmän mukaan relaatiomalli koostuu 13 pääsäännöstä (Kline, Gould & Zanevsky 1999, 6.):

- Relaatiotietokanta käyttää ainoastaan relaatio-ominaisuuksiaan tietokannan hallintaan.
- Kaikki tieto tietokannassa esitetään vain ja ainoastaan yhdellä tavalla (sarakkeina tietoriveissä).
- Kaikkiin tietoihin täytyy päästä käsiksi jollakin tietoyhdistelmällä (esimerkiksi pääavainten avulla).
- Tietokannan jokaisen kentän täytyy voida olla null-arvoinen eli ei mitään.
- Kannan täytyy tukea reaaliaikaista luetteloa, johon auktorisoidut käyttäjät pystyvät tekemään kyselyjä tietokannan kielellä.
- Kannan täytyy tukea vähintään yhtä relaatiokieltä, jolla voidaan tehdä kaikki kannan tietoihin liittyvät tehtävät. Kielellä pitää olla lineaarinen syntaksi ja sitä täytyy voida käyttää muissa ohjelmissa (esimerkiksi SQL).
- Kaikkien teoreettisesti päivitettävien näkymien täytyy olla järjestelmän päivitettävissä.
- Tietokannan tietueiden lisäys, poisto ja päivitys täytyy onnistua myös joukkotasolla.
- Tiedon fyysisen tallennustavan muutoksen ei pidä aiheuttaa sen päällä toimivan sovelluksen muutosta väkisin.

- Tiedon loogisen osan, eli sovelluksen, muutos ei pidä aiheuttaa tietokannan muutosta väkisin.
- Eheysrajoitteet täytyy voida lisätä reaaliaikaiseen luetteloon ja niitä täytyy pystyä tarvittaessa muokkaamaan.
- Tietokantaa hajautettaessa sovellusohjelmien toiminnan ei tarvitse muuttua hajautuksesta johtuen.
- Eheyssääntöjä ei voi ohittaa alemman tason kielillä.

Relaatiotietokannan suurimpana etuna on se, että tiedon fyysinen tallennusjärjestys kiintolevylle muuttuu merkityksettömäksi, koska tietoon päästään aina käsiksi osoittimien avulla. Myös tiedon lisääminen ja poistaminen on helppoa. Relaatiokannan haittana taas on se, että sitä ei pysty kovinkaan helposti jälkeenpäin muokkaamaan muuttamatta koko kannan rakennetta. Relaatiomalli perustuu erillisiin joukkoihin, jotka muistuttavat taulurakennetta. Nämä taas sisältävät erilaisia tietojoukkoja, kuten sarakkeita ja kenttiä. Yksi itsenäinen kenttäjoukko taas koostaa yhden tietorivin. (Kline, Gould & Zanevsky 1999, 4-5.)

Käytännössä jokainen relaatiomallin taulu sisältää vähintään yhden pääavainsarakkeen, jolla taulun jokainen tietorivi yksilöidään. Tämän pääavaimen lisäksi taulu voi sisältää suuren joukon muita tietosarakkeita. Pääavain voidaan koostaa myös useammasta sarakkeesta, mutta yleensä tämä ei ole suositeltavaa. Kaikilla tietosarakkeilla voidaan hakea tietoa, mutta verrattuna muihin sarakkeisiin pääavaimen sarakkeella palautuu aina vain yksi tietorivi. Eri tauluissa olevia rivejä voidaan yhdistää luomalla viiteavaimia (foreign key) toisiin tietokannan tauluihin Nämä relaatiot eri sarakkeiden välillä määritellään tietokannan luonnin yhteydessä. Näistä taulujen relaatioista voidaan hakea tietoa kahdella eri tavalla, joko liitoksena tai unionina. Liitos-ominaisuudella tehty haku palauttaa molemmista tauluista kaikki arvot. Unioni-ominaisuuden haku taas palauttaa vain taulujen yhteiset tiedot. Liitoksien tarkoituksena on estää samojen tietojen tallentamista useaan kertaan eri tauluihin. Pitkän ajan kuluessa vääränlainen varastointi kuluttaa paljon levytilaa ja pidentää kannan hakuaikoja.

Relaatiotietokannan taulujen kuvaamiseen voidaan käyttää esimerkiksi relaatiokaavioita, joissa ilmoitetaan ensin taulun nimi ja sen jälkeen taulun kenttien nimet. Näissä viiteavaimet on yleensä kursivoitu ja pääavaimet alleviivattu.

Henkilö(Etunimi, Sukunimi, Ikä, Pituus, Paino, *Osoite*)

Tietokanta pyritään aina saattamaan ns. kolmanteen normaalimuotoon. Normaalimuotoisuudessa on kolme tasoa ja kanta pyritään aina saattamaan kolmannelle tasolle. Käytännössä tämä tarkoittaa taulujen tarkastelua, kunnes ehtoja rikkovat taulut on muutettu tai pilkottu pienemmiksi tauluiksi niin, että tietokanta saavuttaa sille toivotut ominaisuudet. Ensimmäisessä normaalimuodossa kaikkien sarakkeiden arvot ovat yksimuotoisia, eli esimerkiksi sarakkeen merkkikenttä ei saa olla ”Ford, Nissan, Toyota” vaan sen pitäisi olla pelkästään ”Ford”. Toisessa normaalimuodossa taulussa ei

saa olla tilannetta, jossa vain osa pääavaimesta määrää sarakkeen arvon. Kolmannen normaalimuodon kanta ei vielä lisäksi saa sisältää sarakkeita, jotka ovat riippuvaisia sarakkeista ja jotka eivät ole pääavaimia. (Kline, Gould & Zanevsky 1999, 6-11.)

3.2 SQL-tietokantakieli

SQL (Structured Query Language) on IBM:n kehittämä standardoidu tietokannan kyselykieli, joka on alun perin kehitetty juuri relaatiotietokantojen käsittelyn helpottamiseen. Sen ensimmäinen versio kehitettiin 1970-luvulla ja ensimmäisen kaupallisen version kehitti Oracle 1970-luvun lopussa. Nykyään SQL:stä löytyy paljon eri toteutuksia useilta eri tarjoajilta. (Melton & Simon 2002, 23-25.)

SQL-kieli koostuu selkeistä englanninkielisistä käskysanoista, jotka voidaan jakaa erilaisiin ryhmiin toiminnallisuutensa mukaan. Lauseke on SQL-komennon olennainen osa, joka kertoo mitä tehdään. Predikaatti määrittelee ehdot sille, mitä määrittelyarvoille tehdään. Kyselyllä haetaan tietoa kannasta. Lause tarkoittaa aina yhtä kokonaista puolipilkkuun päättyvää SQL-komentoa. Edellisiä osia sisältävä SQL-komento voisi olla esimerkiksi tällainen:

```
UPDATE Person
SET Town = 'Hämeenlinna'
WHERE Name = 'Harri';
```

Esimerkissä käskyt update, set ja where ovat lausekkeita, eli niillä kerrotaan, mitä ja millä tietorivillä tietoa muokataan. Muita muokkauslausekkeita ovat mm. insert ja delete. Sanat ”Hämeenlinna” ja ”Harri” taas ovat määrittelyjä, jotka sisältävät tietoa, jota verrataan tai asetetaan tietokannan kenttiin. Predikaatteja esimerkissä ovat ”Town = ’Hämeenlinna’” ja ”Name = ’Harri’”. Koko kolmen rivin komentoa taas kutsutaan lauseeksi, koska se päättyy puolipilkkuun. Sana ”Person” lauseessa kuvaa tietokannan taulua ja sanat ”Town” sekä ”Name” kuvaavat tietokannan taulun kenttiä.

Edellisen esimerkin lauseen tarkoituksena oli muokata tietoa kannassa. Yleisemmin riittää kuitenkin pelkkä tiedon haku, joka onnistuu tekemällä SQL-kyselyn kantaan. Myös kysely koostuu useammasta eri elementistä, joilla vaikutetaan kyselyn palauttamaan tietoon. Kysely kantaan voisi siis olla esimerkiksi tällainen:

```
SELECT * FROM Person
WHERE Name = 'Harri'
ORDER BY Town;
```

Edellinen esimerkkikysely palauttaa Person-taulusta kaikki tietorivit, joiden Name-sarakkeen arvo on Harri. Tämän jälkeen saadut arvot vielä järjestetään Town-sarakkeen arvojen mukaan aakkosjärjestykseen. Mikäli kyseessä olisi numeroita sisältävä sarake, arvot järjestettäisiin pienimmästä suurimpaan.

SQL tukee myös ns. transkatioiden kontrollointikomentoja, joilla voidaan varmistaa tiedon eheys kannassa tallennettaessa tai poistettaessa tietoa. Transaktiot ovat siis käytännössä tiedon muutoksia kannassa. Esimerkissä poistetaan tietoa taulusta hyödyntämällä transaktion kontrollointia:

```
START TRANSACTION;
DELETE FROM Person
  WHERE Name = 'Harri';
COMMIT;
```

SQL-komennoilla tehdään myös muutoksia tietokannan rakenteeseen, eli poistetaan tai muokataan tauluja jne. Esimerkki taulun luomisesta:

```
CREATE TABLE Person
{
  Firstname  VARCHAR(15);
  Lastname   VARCHAR(25);
  Age        INT;
};
```

Komento luo taulun Person ja asettaa siihen sarakkeet Firstname, Lastname ja Age sekä antaa näille tietotyypit. Varchar tarkoittaa vapaasti kirjoitettavaa tekstiä, jonka pituus määräytyy suluissa, ja int tarkoittaa kokonaislukua.

SQL sisältää useita erilaisia tietotyyppejä. Nämä toimivat alkioina muuttujille, joiden avulla määritellään, millaista tietoa kyseinen muuttuja saa sisältää ja minkä pituinen tai kokoinen muuttujaan määriteltävä arvo saa olla. Erilaisia numeraalisia tietotyyppejä ovat esimerkiksi int, double ja single. Tekstiä sisältävä tietotyyppi on varchar. Päiväyksiä sisältävä tietotyyppi taas on nimeltä date.

SQL:n päälle on tehty useita erilaisia toteutuksia eri palveluntarjoajilta. Näissä toteutuksissa on sekä hyviä että huonoja puolia riippuen siitä, mihin ne ovat pääasiassa tarkoitettu. Jotkut toteutukset on tehty yksinkertaisiksi käyttää esimerkiksi selkeän graafisen käyttöliittymän avulla. Toisten tarkoituksena taas on olla tehokkaita kannan koon kasvaessa isoksi. Microsoftin tarjoamia SQL-toteutuksia ovat Microsoft Access, jolla voidaan tehdä melko yksinkertaisia Access-tyyppisiä kantoja ja joita voidaan helposti kontrolloida graafisella käyttöliittymällä tai SQL-komentojen avulla. Toinen Microsoftin toteutus on Microsoft Query, joka tarjotaan Microsoftin omien kehitystyökalujen, esimerkiksi Visual Basicin mukana. Oracle taas tarjoaa yksityiskäyttäjille Personal Oracle tietokantapalvelua, josta saa ainakin kokeiluversion ilmaiseen käyttöön. Oracle tarjoaa myös yrityksille huomattavasti robustimpaa versiota kantaratkaisustaan. (Melton & Simon 2002, 2-3.)

3.3 Microsoft SQL Server

Seuraavat kappaleet on lainattu SQL Server Books Online (2010) -lähteestä. SQL Server on Microsoftin tekemä tietokantapalvelin, jonka tällä hetkellä uusin versio on 2008 R2. Microsoft SQL Server on alun perin kehitetty

Sybasen ja Microsoftin yhteistyönä. Ensimmäinen versio SQL Serveristä julkaistiin vuonna 1989 OS/2 käyttöjärjestelmälle. Myös vielä seuraava versio julkaistiin OS/2:lle, jonka jälkeen Microsoft lopetti yhteistyön Sybasen kanssa ja kehitys siirtyi Windows NT -puolelle. SQL Server käyttää kyselykielenään perus-SQL:ää sekä siitä Microsoftin ja Sybasen muokkaamaa T-SQL:ää. SQL Serveristä jaellaan useampaan eri versiota eri käyttäjäkunnille.

Koska SQL Server on tietokantapalvelin, täytyy siinä olla rajapinta, jolla muut ohjelmat kommunikoivat tietokannan kanssa. Tämä on toteutettu ns. tabular data stream (TDS) -protokollalla. TDS:llä tietokannasta siirrettävästä ja siirretystä tiedosta luodaan käytettyyn siirtoprotokollaan sopivia paketteja. Esimerkiksi TCP/IP protokollalla siirrettäessä paketit muutetaan TCP/IP:lle sopiviksi paketeiksi. Paketin tieto siirtyy joko XML tai relaatiotietokannan tietokokoelmana riippuen siitä, onko XML tiedonsiirto määritelty palvelimelle. Etuna XML-pohjaisessa siirrossa on se että XML-tiedostot ovat selkokielisiä ja niitä voidaan lukea esimerkiksi selaimessa. TDS:n lähettämien pakettien kokoa voidaan muokata tarvittaessa, mutta peruskoko on 4 kilotavua.

SQL Server voidaan jakaa tietokantakoneeseen sekä tietokantapalveluihin. Tietokantakone on SQL Serverin ydinpalvelu, joka huolehtii tiedon hallintaan liittyvistä operaatioista kuten tallentamisesta, prosessoinnista ja tiedon varmentamisesta. Tietokantakoneella siis luodaan käytettävät tietokannat sekä niiden taulut ja hallinnoidaan niitä. Tietokantaa voidaan hallita SQL Serverin Management Studiolla, joka on käytännössä graafinen käyttöliittymä kannan hallinnointiin. Tietokannan hallintaan voidaan käyttää myös SQLCMD:a, joka taas on komentorivipohjainen tietokannan hallintatyökalu.

SQL Serverin tietokantapalveluita ovat analysointipalvelut, integrointipalvelu, replikointipalvelu, raportointipalvelu ja erityisesti ohjelmoijille suunnattu Service Broker -palvelu. Analysointipalveluita ovat moniulotteinen tietoanalysointi ja tietolouhinta. Moniulotteisella tietopalvelulla voidaan mahdollistaa tiedon haku useammasta lähteestä, esimerkiksi useista relaatiotietokannoista, yhdeksi määritellyksi tietokokonaisuudeksi. Tällä palvelulla voidaan siis helpommin analysoida suuria tietomääriä useista eri tietolähteistä. Tietolouhintapalvelulla voidaan luoda algoritmeja sekä tiedonlouhintamalleja, joilla tietolouhintaa toteutetaan. SQL Server tarjoaa myös oman DMX- kielen, jolla tietolouhintamalleja voidaan toteuttaa.

Integrointipalvelulla voidaan yhdistää tietoa useista eri tietolähteistä ja muuntaa sitä edelleen toisiin palveluihin tai lähettää esimerkiksi sähköpostiin. Replikointipalvelulla voidaan kopioida kokonaisia tietokantaobjekteja kannoista toisiin ja synkronoida kannat sen jälkeen toisiinsa. Replikoinnin avulla voidaan jakaa tieto useampiin eri tietolähteisiin. Replikointipalvelulla voidaan replikoida myös tietokantatransaktioita.

Raportointipalveluilla voidaan luoda raportteja tietokannan tiedoista. Sillä voidaan luoda, muokata ja hallita raportteja sekä kustomoida raportointitoimintoja. Raportointipalveluilla voidaan siis generoida raportteja annettujen parametrien mukaisesti. Palvelua hallinnoidaan web-käyttöliittymän avulla. Raportointipalveluilla voidaan tuottaa raportteja kannasta useissa eri tiedostomuodoissa, esimerkiksi XML-muodossa.

Service Broker -palvelu tarjoaa SQL Server -tietokantakoneelle natiivin tuen sovellusten viestintään. Tämä helpottaa sovelluskehitystä sekä rajapintojen luontia eri kantojen välillä. Tietokantojen työtaakan jakaminen on helpompaa tämän palvelun avulla koska kantojen välistä sovellusviestintää ei tarvitse itse ohjelmoida. Service Broker kommunikoi toisten sovellusten kanssa TCP/IP -protokollan avulla ja mahdollistaa eri komponenttien synkronoinnin viestien välityksellä. (SQL Server Books Online 2010.)

3.4 Microsoft AX:n tietokanta

Microsoft AX käyttää tietokantanaan relaatiotietokantaa, joka voi toimia joko Microsoftin omalla SQL Serverillä tai Oraclen tietokantapalvelimella. Se sisältää kaiken standardin relaatiotietokannan toiminnallisuuden ja joitakin AX:n mukana tulevia ominaisuuksia, kuten historiatietojen tarkastelun sekä raporttien luomisen. Lisäksi AX kannassa voidaan erotella eri yritysten tiedot samoissa tauluissa niin, että ohjelmalogiikka toimii samalla tavalla riippumatta siitä, mikä yritys AX:ssä on valittuna. AX tallentaa kaikkien yritysten tiedot samoihin tauluihin AOS:ssa (Application Object Server) eli tietokantapalvelimella, mutta piilottaa muiden kuin valittuna oleva yrityksen tiedot. Muiden yritysten tietoja ei voi myöskään ilman erityiskomentoja valita, poistaa tai muokata. Tällä tavalla estetään muiden yritysten tietojen muuttaminen vahingossa. (Microsoft Dynamics AX 4 Tables, Maps and Views 2010.)

3.4.1 Tietokantataulut

Seuraavat kappaleet on lainattu Microsoft Dynamics AX 4 Tables, Maps and Views (2010) -lähteestä. Tietokantatauluja käytetään tiedon tallentamiseen AX:ssä, eli ne ovat käytännössä taustalla olevan relaatiokannan tauluja, joihin pätevät relaatiokannan lisäys-, poisto- ja muokkauskomennot. AX:n kaikki taulut löytyvät AOT:sta (Application Object Tree) polusta data dictionary/tables. Jokainen taulu sisältää samat peruselementit, eli kentät (fields), kenttäryhmät (field groups), indeksit (indexes), relaatiot (relations), poistotoiminnot (delete actions) ja metodit (methods).

Kentät ovat tietokannan kenttiä, joihin tieto tallennetaan määrätyssä muodossa. Kenttään tallennetun tiedon muodon voi määritellä joko numeroksi, tekstiksi, päiväykseksi ym. Kentälle voidaan myös antaa muita ominaisuuksia kuten pakollisuus ja perusarvo. Jokainen uusi taulu sisältää tietyt järjestelmäkentät (system fields), jotka AX luo automaattisesti kun

uusi taulu luodaan. Näitä kenttiä ei voi poistaa tietokantataulusta. Niiden tarkoitus on määritellä tiedolle yksilöivä tunniste ja antaa tietoa siitä, kuka ja mikä kyseistä tietueriviä on muokannut.

Kenttäryhmien tarkoituksena on yhdistää loogisesti toisiinsa kuuluvat kentät yhdeksi ryhmäksi. Käytännössä jokaisen tietokannan kentän pitäisi kuulua kenttäryhmään. Tällä tavalla niitä on huomattavasti helpompi käyttää näkymissä ja raporteissa. Kaikki yhdessä kenttäryhmässä olevat kentät voidaan kerralla yhdistää näkymässä tai raportissa.

Indeksit ovat taulukohtaisia tietokantarakenteita, joiden tarkoitus on nopeuttaa hakua tietokannasta. Niillä voidaan myös määrittää tietoriveille yksilöiviä tietuekenttiä. AX luo automaattisesti järjestelmäindeksin niille tietueriveille, joille käyttäjä ei ole luonut mitään indeksiä. Tämä indeksi perustuu recId ja dataAreaId -kenttiin. Käytännössä siis indeksit ovat yhdestä tai useammasta kentästä koostuvia tietueita, joilla voidaan yksilöidä tietorivejä ja nopeuttaa hakua kannasta. Indeksin ei tarvitse välttämättä olla uniikki. Indeksiä voidaan käyttää myös tiedon haun optimointiin.

Relaatioissa voidaan määritellä taulujen välisiä relaatioita. Taulujen relaatioilla pakotetaan kahden tai useamman taulun relatiiviset tiedot vastaaviksi. Relaatioiden avulla voidaan tehdä esimerkiksi näkymien (form) kenttiin hakunappi, jolla haetaan lista toisessa taulussa olevista tiedoista.

Poistotoiminnoilla määritellään mitä tapahtuu, kun taulun tietuerivi poistetaan, jotta kannan yhtenäisyys säilyy. Poistotoimintoja luodaan yleensä sellaisiin tauluihin, joista on relaatioita muihin tauluihin. Tällöin poistotoiminnalla voidaan määritellä mitä tapahtuu toisessa taulussa olevalle tietueriville, kun relaation toisessa taulussa poistetaan rivi. (Microsoft Dynamics AX 4 Tables, Maps and Views 2010.)

3.4.2 Taulunäkymät

Seuraavat kappaleet on lainattu Microsoft Dynamics AX 4 Tables, Maps and Views (2010) -lähteestä. Taulunäkymä on virtuaalinen taulu, jonka sisältö määritellään tauluun tai tauluihin tehdyllä tietokantakyselyllä. Tietokantanäkymällä on kenttiä ja rivejä, mutta näkymän tieto tulee aina kyselyn palauttamista tiedoista. Tietokantanäkymä luodaan aina dynaamisesti uudestaan kun se avataan. Sitä ei siis koskaan tallenneta tietokantatauluksi. Tietokantanäkymiä voidaan käyttää samalla tavalla kuin tauluja eli niiden tietoja voidaan hyödyntää esimerkiksi raporteissa. Tietokantanäkymän kautta ei voida kuitenkaan päivittää kannan taulun tietoja eli niitä voidaan vain lukea.

Tietokantanäkymän etuja ovat niissä oleva kohdistettu tieto, räätälöity tieto sekä niiden optimointi. Tietokantanäkymään voidaan tuoda halutessa vain ne tietokantataulujen kentät, joista tietoa tarvitaan kyseisessä tilanteessa. Tiedot voidaan myös räätälöidä esimerkiksi niin, että tietoa haetaan useammasta taulusta, mutta vain tiettyjä kenttiä. Optimointi taas tapahtuu

kun haetaan vain ne kentät, joita käyttäjä tarvitsee. Tällä tavalla säästetään tietokantapalvelimen kuormaa. Tietokantanäkymään voidaan siis määritellä halutut taulut ja kentät, mahdolliset tietosuodattimet, taulujen väliset relaatiot sekä tarvittaessa aggregaatti eli summakentät.

Taulunäkymät koostuvat neljästä peruselementistä, jotka ovat metatiedot (metadata), kentät, kenttäryhmät ja metodit. Metatiedoilla määritellään mitä tietoa näkymään tullaan hakemaan eri tauluista. Se sisältää siis tietolähteet (datasources), joista tieto haetaan. Metatiedot sisältävät myös tietoalueen (range), jonka perusteella tieto haetaan kannan tauluista. (Microsoft Dynamics AX 4 Tables, Maps and Views 2010.)

3.4.3 Assosiaatiotaulut

Seuraavat kappaleet on lainattu Microsoft Dynamics AX 4 Tables, Maps and Views (2010) -lähteestä. Assosiaatiotauluilla määritellään X++-elementtejä, joilla voidaan viitata tietokannan kenttiin ajon aikana. Assosiaatiotauluilla voidaan määritellä assosiaatiotaulun kentän ja yhden tai useamman tietokantataulun kentän välille yhteys. Assosiaatiotaulun kenttä voidaan siis nimetä erilaiseksi kuin kenttä, tai kentät, johon kenttä viittaa. Samaan tauluun voidaan viitata useammalla kuin yhdellä assosiaatiotaululla. Tällöin yleensä eri assosiaatiotauluilla viitataan eri tietokannan kenttiin tai kenttäryhmiin.

Assosiaatiotaulujen tarkoituksena ja hyötynä on se, että ne yksinkertaistavat ja helpottavat taulujen kenttien muokkaamista. Niillä voidaan siis luoda yksi liittymä, jonka avulla muokataan useita kenttiä tauluissa. Tämä tarkoittaa sitä, että oliota, joka viittaa johonkin taulun kenttään, voidaan käyttää useammassa taulussa muuttamatta itse taulujen kenttänimiä. Assosiaatiotaulut myös helpottavat ohjelmakoodin yhtenäisyyden säilyttämistä, koska eri taulujen erinimisiin kenttiin voidaan viitata samalla tavalla. Niillä voidaan myös helpottaa ohjelmakoodin uudelleen käyttöä assosiaatiotaulujen metodien avulla, jotka ajetaan assosiaatiotaulujen kenttiä vasten. Tämä tarkoittaa sitä, että eri metodeja ei tarvitse kopioida moneen kertaa eri tauluille.

Esimerkkinä assosiaatiotauluista on AX:n address-assosiaatiotaulu, jolla voidaan käsitellä tietoja kahdessa eri taulussa. Tällä tavalla voidaan siis viitata kahteen erinimiseen kenttään samalla nimellä ja käyttäen samoja metodeja. Tässä assosiaatiotaulussa kenttä Address viittaa kenttiin Address ja ToAddress tauluissa Address ja CustVendTransportPointLine.

Assosiaatiotaulut koostuvat neljästä peruselementistä: kentistä, kenttäryhmistä, assosiaatioista (mappings) ja metodeista. Assosiaatioilla määritellään se, mihin taulujen kenttiin assosiaatiotaulun kentät viittaavat. Niillä myös määritellään taulut, johon assosiaatiotaulu viittaa. Assosiaatiotaulujen kenttien täytyy olla samaa tietotyyppiä kuin taulun kentän, johon se viittaa. (Microsoft Dynamics AX 4 Tables, Maps and Views 2010.)

4 TIEDON IMPLEMENTOINTITEKNIIKAT MICROSOFT AX -JÄRJESTELMÄSSÄ

Tässä luvussa esitellään AX:n kehitysympäristö sekä muut tiedonsiirtoon liittyvät oleelliset asiat. Luvun tarkoituksena on antaa kuva siitä mitä osa-alueita tiedon implementointi AX:n käsittää ja millaista osaamista se vaatii. AX:n sovelluskehitysympäristö koostuu kolmesta pääosasta, MorphX ohjelmointiympäristöstä, X++-ohjelmointikielestä sekä AOT-kehitysvalikosta. Lisäksi kappaleessa käydään läpi tiedonsiirtoon liittyviä AX-tekniikoita.

4.1 Morphx

Microsoft AX Dynamicsin graafinen ohjelmointiympäristö eli IDE on nimeltään MorphX. Se on AX:n integroitu kehitysympäristö, joka koostuu AOT:stä (Application Object Tree), X++-ohjelmointikielestä sekä muista työkaluista, joilla saadaan tietoa järjestelmästä. Kaikki AX:n mukana tulevat moduulit on toteutettu MorphX:n avulla. Näin ollen mikäli yritys on lisensoinut AX:n lähdekoodin, Microsoftin kaikkia valmiita moduuleita voi halutessaan muokata oman yrityksen tarpeita vastaavaksi. ERP-paketteihin AX on erittäin helposti muokattavissa. MorphX tukee Microsoft SQL Server- ja Oracle-tietokantaa. Tietokannan valinnalla ei tosin ole suurta merkitystä MORPHX:n kannalta, sillä tietokantaan liittyvät asiat jäävät piiloon Axaptan ytimen (kernel) taakse. (Andreasen 2006, 19.)

4.2 X++

X++ on MorphX-kehitysympäristön olio-pohjainen ohjelmointikieli, joka pohjautuu C++-kieleen. X++ on kehitetty AX:ää varten helpottamaan tietokannan muokkausta. X++:lla ei tarvitse erikseen hallinnoida tietolähteitä vaan tietokantakyselyt voidaan kirjoittaa suoraan koodiin SQL-tyyppisesti. X++-koodia ei ole mahdollista piilottaa, vaan kaikki itse tehdyt luokat, metodit jne. näkyvät vapaasti kaikille käyttäjille. Myös kaikki alkuperäinen AX:n mukana tuleva X++-koodi on täysin näkyvissä ja vapaasti muokattavissa sekä kopioitavissa. Tästä on omissa ohjelmointitoteutuksissa suurta hyötyä. Vaikka X++ on olio-pohjainen ohjelmointikieli, sillä voidaan ainoastaan periä luokkia ja laajennettuja tietotyyppejä (extended datatypes). Syntaksiltaan X++ muistuttaa esimerkiksi Javaa. (Andreasen 2006, 45.)

Tiedonsiirrossa X++-syntaksin hallitseminen on tarpeellista, sillä AX sisältää monia tauluja, joissa pelkkä tiedon siirto esim. Excel-pohjalla ei riitä. Normaalissa siirrossa järjestelmä ei esimerkiksi automaattisesti generoi numerosarjoista tulevia yksilöiviä tunnistetietoja, kuten rakenne- ja reitti-

tunnisteita. Myöskään relaatiot eivät aina synny halutulla tavalla. Näitä ongelmia voi paikata joko siirron jälkeen luomalla lyhyen koodinpätkän, jolla muokataan siirrettyjä tietoja, tai muokkaamalla siirron aikana ajettavaa ns. konversiokoodia, joka voidaan määrittää taulun määrittelyryhmään (definition group). Konversiokoodi ajetaan siirron yhteydessä jokaisen taulun uuden tietorivin kohdalla. Koska kaikki X++-luokat ovat avoimia, voidaan tarvittaessa myös siirrosta vastaavien luokkien koodia muokata. Tässä kannattaa kuitenkin olla varovainen, sillä luokan metodeita väärin muokkaamalla saattaa onnistua rikkomaan koko luokan toiminnan.

X++-ohjelmoinnissa tärkeintä osaa esittävät erilaiset tietokannan muokkaukseen liittyvät komennot. Syntaksiltaan nämä komennot muistuttavat hyvin paljon aikaisemmin esitettyä SQL-syntaksia. X++-tietokantakäskyt luovatkin itse asiassa normaaleja SQL-komentoja kantaan. X++-koodiin onkin mahdollista upottaa myös normaaleja SQL-komentoja. Tämä ei kuitenkaan ole yleensä tarpeellista, sillä X++-tietokantamuokkauskomentojen kirjoittaminen on huomattavasti yksinkertaisempaa ja nopeampaa.

Hyvin yksinkertainen X++-metodi voisi olla esimerkiksi tällainen:

```
Public PersonTable returnPerson(str _name)
{
PersonTable Person;
;
Select * from Person where Person.name == _name;
Return Person;
}
```

Alussa oleva Public-komento määrittää metodin julkiseksi eli sitä voidaan kutsua mistä tahansa muualtakin kuin vain luokan sisältä. Seuraavana oleva PersonTable-sana tarkoittaa palautettavaa tietotyyppiä, tässä tapauksessa metodi palauttaa kuvitteellisen PersonTable-taulun. Metodin kutsumanimi taas on returnPerson ja sille täytyy antaa parametri _name, jonka avulla suoritetaan select-komento. Seuraavaksi määritellään muuttujaan Person PersonTable-taulu, jolloin ohjelma alustaa Person-muuttujan PersonTableksi, eli periaatteessa puskuriksi, johon tehdään tauluun tehtävät haut ja muutokset. Select-lause on hyvin samankaltainen kuin SQL:ssä. Lopuksi metodi palauttaa Person-muuttujan.

4.3 AOT

AOT eli Application Object Tree on AX:n kehitysvalikko. Järjestelmän kaikki objektit, kuten näkymät, taulut, kyselyt, luokat, valikot ym. löytyvät AOT:sta. AOT:n sisältö näkyy rakennemaisena puunäkymänä, jossa kaikki objektit löytyvät oman otsikkonsa alta. (Andreasen 2006, 19.)

AX:ssa kaikki objektit varastoidaan eri kerroksille (layers), joita löytyy yhteensä 16. Kerrosten tarkoituksena on erotella Microsoftin, AX:n omistavan yrityksen, sekä kolmansien osapuolien tekemät objektit ja muutokset niihin. Ylempien kerrosten muutokset näkyvät aina

ensimmäisenä AX:ssä. Alempia kerroksia ei kuitenkaan koskaan ylikirjoiteta. Tämä tarkoittaa sitä, että tuhoamalla ylemmän kerroksen muutokset voidaan järjestelmä aina palauttaa alkuperäiseen malliinsa. Alempiin kerroksiin tehdyt muutokset taas siirtyvät automaattisesti ylempiin kerroksiin. (Andreasen 2006, 20-22.)

Minkä tahansa objektin ominaisuuksia (property) voidaan muokata AOT:n kautta. Myös metodeja voidaan muokata AOT:n kautta X++-editorilla. Objekteja voidaan siirtää AOT:sta toiseen import- ja export-toiminnolla. Tätä voidaan käyttää esimerkiksi siirrettäessä objekteja kehitysympäristöstä tuotantoympäristöön. (Andreasen 2006, 23.)

AOT on tiedonsiirron kannalta miltei välttämätön työkalu. Sen kautta pystytään helpoiten määrittelemään, mihin tauluihin tietoa täytyy siirtää, jotta tieto näkyy tietyssä näkymässä. Esimerkiksi tuotetietojen perustietotaulua vietäessä voidaan AOT:ta hyödyntää katsomalla, mitä tietolähteitä (datasources) perustietotaulun näkymällä on. Tietolähteistä taas voidaan selvittää kaikki taustalla olevat taulut ja niiden relaatiot toisiin tauluihin. Koska AOT-tieto on indeksoitu, voidaan siihen myös tehdä hakuja, mikä helpottaa esimerkiksi metodien etsimistä.

Kuva 2. *Application Object Tree.*

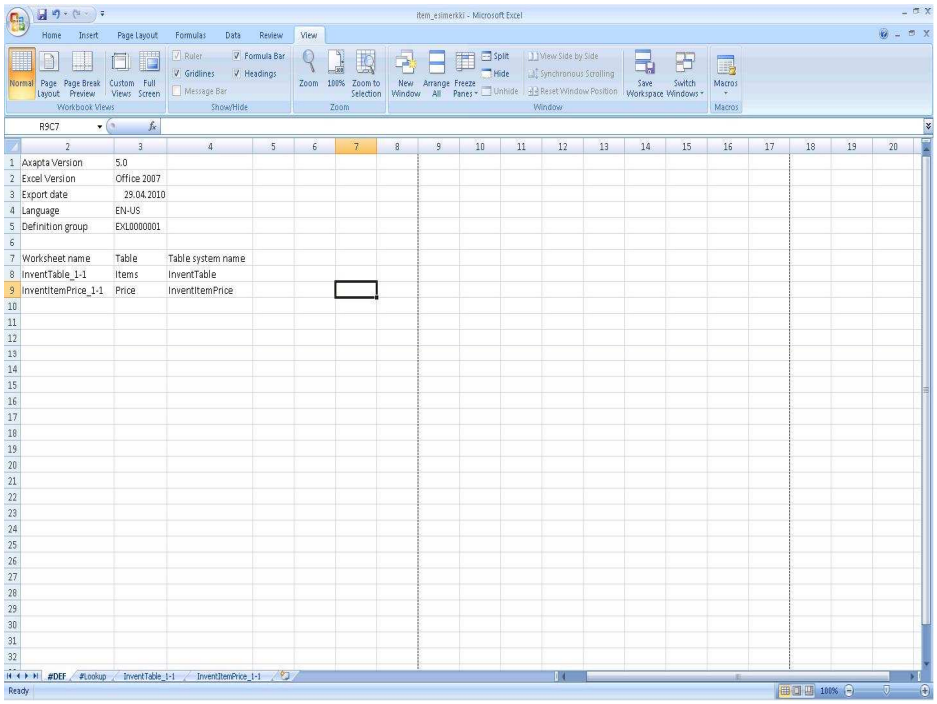
Esimerkkikuvassa on AOT:n kaikki yläsolmut (node) kuvattuna. Näiden solmujen takaa löytyy useita erilaisia alisolmuja. Ylärivin painikkeista oleellisin on ominaisuudet (properties) -painike. Sen takaa saadaan lisätietoa

valitusta AOT-objektista, esimerkiksi näkymistä saadaan muokattua erilaisia näkymien ominaisuuksia.

4.4 Siirtotiedostomuodot ja niiden erot

AX:stä voidaan tuoda ja AX:n voidaan viedä tietoa kolmessa eri muodossa. Nämä muodot ovat binääritiedosto, comma-tiedosto ja Excel-tiedosto. Binääritiedostoa käytetään pääasiassa tiedon varmentamiseen. Binääritiedostot kirjoitetaan nopeasti ja koska ne ovat pakattuja, niihin voidaan tallentaa suuri määrä tietoa ilman, että tiedostokoko kasvaa liian suureksi. Niitä ei voida kuitenkaan käsitellä muualla kuin AX:ssä. Binääritiedostoilla voidaan myös esimerkiksi kopioida yritystietoja nopeasti. Tämä sopii hyvin tilanteisiin, joissa tietoa ei tarvitse millään lailla muuttaa. Comma- ja Excel-tiedostot ovat tarkoitettu tietojen tuomiseen tarkastelua ja muokkaamista varten taulukkolaskentaohjelmaan. Niiden erona on se, että AX:n luomaa Excel-tiedostoa voidaan käyttää myös tiedon viemiseen järjestelmään. AX luo tarvittavat määritykset Excel-pohjaan, jonka perusteella Microsoft Office Excel Import -toiminnolla voidaan siirtää tietoa järjestelmään.

Uusia tietoja järjestelmään siirrettäessä käytetään pääasiassa Excel-siirtopohjia. Ne rakentuvat pääsivusta, jossa määritellään taulut, jotka järjestelmään halutaan viedä. Pääsivulla voidaan lisäksi määritellä, missä järjestyksessä taulut halutaan siirtää järjestelmään. Siirtopohjan muut sivut merkitsevät siirtopohjaan määriteltyjä tauluja. Jokaiselta sivulta löytyvät lisäksi kyseiseen tauluun liittyvät kentät. Siirtopohjan tekovaiheessa voidaan määritellä, mitkä taulut ja valittujen taulujen kentät siirtopohjaan halutaan viedä.



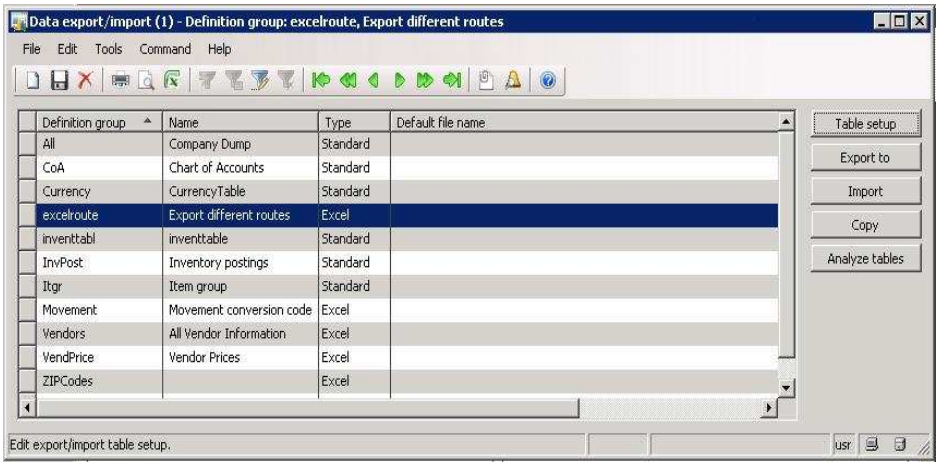
Kuva 3. *Excel siirtopohja.*

Esimerkkikuvassa valittuna oleva sivu on siirtopohjan pääsivu jonka AX nimeää #Def -nimellä. Tällä sivulla näkyvät tietoina ohjelmistojen versiot, siirtopohjan luontipäivä ja kieli sekä sen määrittelyryhmä. Kolme alinta riviä kertovat taulun sivunimen Excelissä, taulun nimen sekä taulun järjestelmänimen. Tärkeimmät tiedot ovat taulun sivunimi ja taulun järjestelmänimi, näitä tietoja ei pidä muokata. #Lookup -sivulla määritellä mahdolliset monivalintakenttien arvot. Sivut Inventtable-1-1 ja InventItemPrice-1-2 ovat siirtopohjaan valittujen taulujen nimiä.

4.5 Hallintamoduuli ja tiedonsiirto

Administration module eli hallintamoduuli on yksi AX:n mukana tulevista perusmoduuleista. Se on käytännössä hallintamoduuli järjestelmän pääkäyttäjille. Sen avulla voidaan valvoa ja muokata AX toimintaa käyttäjä- ja yritystasolla. Yksi tärkeimmistä tehtävistä, mitä tässä moduulissa voidaan tehdä, on käyttöoikeuksien hallinta. Hallintamoduulin näkymissä voidaan luoda uusia käyttäjiä ja ryhmiä ja määrittää näille käyttöoikeuksia. Käyttöoikeuksien hallinta AX:ssa onnistuu hyvin pikkutarkasti, jopa eri taulujen kenttiin voidaan asettaa eri käyttäjille erilaisia oikeuksia. Tätä kautta voidaan myös määrittää käyttäjille erilaisia käyttäjäkohtaisia valintoja, kuten nimi, toimialue, kieli jne. Myös erilaisia käyttäjäkohtaisia valintoja löytyy runsaasti. Tätä kautta voidaan myös valvoa, onko käyttäjä kirjautuneena järjestelmään ja tarvittaessa esimerkiksi päättää käyttäjän istunto. Myös yritysten hallinnointi tehdään hallintamoduulin kautta. Yrityksiä voidaan poistaa, lisätä ja muokata tätä kautta. Moduuli sisältää myös erilaisia tietokannan seurantaan liittyviä raportteja, joiden avulla voidaan seurata mm. valitun yrityksen tietokannan kokoa sekä tehtyjä SQL-transaktioita.

Hallintamoduulin kautta toteutetaan myös tiedonsiirto järjestelmään eri tiedonsiirtopohjien avulla. Tämä tehdään definition groups –näkymästä, joka löytyy periodic-otsikon alta. Näkymän kautta tehdään käytännössä kaikki tiedonsiirtoon liittyvät määrittelyt ja toteutukset. Sen kautta määritellään halutut taulut eri määrittelyryhmiin (definition groups) ja kerrotaan, minkä tyyppinen siirtotiedosto on kyseessä. Määrittelyryhmän luomisen jälkeen table setup -painikkeen takaa valitaan siirtoa varten halutut taulut sekä mahdolliset muut toimenpiteet, joita siirron aikana halutaan tehdä. Kun määrittelyryhmä on säädetty kokonaan valmiiksi, voidaan sen avulla joko siirtää tietoa ulos järjestelmästä tai sisään järjestelmään.



Kuva 4. *Definition group -näkymä.*

Excel-siirtopohjat voidaan luoda myös siirtopohjavelhon avulla, joka vaiheistaa pohjan luontivaiheet selkeästi ja antaa ohjeita luomisen aikana. Käytännössä suuri osa normaaleista taulusiirroista voidaan tehdä velhon avulla luoduilla siirtopohjilla ilman, että definition groups -näkymään tarvitsee edes siirtyä. Monimutkaisempia siirtoja tehdessä tämä näkymä on kuitenkin tarpeellinen, sillä table setup -painikkeen takaa määritettävällä taulukohtaisella konversiokoodilla saatetaan joutua esimerkiksi luomaan ajon aikana relaatiot eri taulujen välille. Siirtopohjavelho löytyy saman periodic-otsikon alta kohdasta template wizard.

Käytännössä Excel-siirtopohjilla tehdyistä siirroista huolehtii AOT:sta löytyvä SysDataImportExcel-luokka. Kun tiedonsiirto aloitetaan, luokassa olevat metodit käyvät läpi Excel-siirtopohjan sivut ja niissä olevat tietorivit. Jokaista tietoriviä kohden ajetaan tauluun doInsert-komento, joka siirtää tietorivin tiedot taulun riville määriteltyihin kenttiin. Koska tiedot ajetaan tauluun doInsert komennolla, on tärkeää varmistaa, että tieto on oikeassa muodossa heti siirrettäessä. Kyseinen komento ohittaa oikeellisuustarkistukset eli kenttään voidaan periaatteessa syöttää mitä tahansa. Tällöin esimerkiksi sama yksilöivä avain voidaan syöttää useaan kertaan usealle riville. Myös Excelissä olevien kenttämääritysten tietotyypit pitää tarkistaa oikeiksi. Mikäli tietokannan taulussa oleva kenttä on määritetty string- eli tekstimuotoiseksi, täytyy myös Excelissä olevan vastaavan kentän tiedon olla tekstimuodossa. Jos samassa kentässä oleva tieto on esimerkiksi määritelty numeroksi, jättää Excel-siirto kyseisen kentän yleensä täysin tyhjäksi ilman mitään virheviestiä. Yleinen virhe siirroissa on siirtää numeromuotoinen id-koodi kenttään, joka on määritelty tekstimuotoiseksi. Tällöin kaikki id-koodikentät jäävät tyhjäksi, mutta muut tiedot siirtyvät. Tämä tarkoittaa käytännössä sitä, että kaikki siirretyt tiedot täytyy poistaa ja siirtää uudelleen.

Excel-pohjille löytyy oma import-näkymänsä, joka on käytännössä supistettu versio normaalista import-näkymästä. Definition group -kenttään voidaan halutessa antaa määrittelyryhmä, jonka mukaan tieto ajetaan kantaan. File name -kenttään määritetään siirrettävän Excel-tiedoston nimi. Advanced-välilehdellä on tärkeää määrittää import rule -kenttä. Tämä

määrittää sen, miten tieto siirretään kantaan, eli tyhjennetäänkö kohde taulu ennen siirtoa, siirretäänkö vain uudet rivit vai siirretäänkö uudet ja päivitetään jo kannasta löytyvät tiedot.

5 TUOTETIEDON IMPLEMENTOINTI MICROSOFT AX -JÄRJESTELMÄÄN

Toiminnanohjausjärjestelmän oleellisimpia osia on tuotetieto, joka koostuu AX:ssä useasta eri taulusta. Luvussa esitellään taulujen määrittely toimivaa tuotetietoa varten, joka tarkoittaa siis perustietotaulujen, sekä reitti- ja rakennetaulujen määrittelyä. Tämän jälkeen suoritetaan itse tiedon siirto Excel-pohjien avulla. Lisäksi luvussa käydään läpi tiedon oikeellisuuden tarkistus eli validointi.

5.1 Perustietotaulujen määrittely

Perustietotaulujen määrittely on helpointa aloittaa siirtymällä item details -näkymään, joka löytyy inventory management -moduulista. Tässä näkymässä voidaan luoda ja poistaa tuotteita sekä muokata tuotteiden ominaisuuksia. Luomalla uuden tuotteen tässä näkymässä näkee helposti taulussa määritellyt pakolliset kentät punaisella alleviivattuna. Nämä kentät näkyvät myös Excel-siirtopohjissa erikseen väritettynä. AX ei anna tallentaa tuotetta tietokantaan ennen kuin vähintään kaikki pakolliset kentät ovat määritetty, tämä tarkoittaa myös muissa tauluissa olevia pakollisia kenttiä.

Tarkastelemalla näkymää on kuitenkin vaikea määritellä, mikä kenttä kuuluu mihinkin tauluun. Siksi kannattaakin avata Application object tree ja etsiä Forms -solmu (node), jonka alta löytyvät kaikki AX:n näkymät. Item details -näkymä löytyy solmusta nimellä inventtable. Solmun alta löytyy tietolähteet-alisolmu, jossa määritellään eri taulut, joista kyseinen näkymä hakee tietoa ja jonne tallennetaan tietoja. Tietolähteen taulun saa selville avaamaalla properties-ikkunan ja tarkistamalla tietolähteen nimen alta taulun järjestelmänimen. Tässä tapauksessa kaikki tietolähteiden taulut ovat oleellisia, jotta tuotetiedot saadaan toimimaan oikein järjestelmässä.

Taulukko 1. *Tuotetaulut.*

Taulun nimi	**Selite**
Inventtable	Sisältää tuotedataan liittyvät perustiedot. Tämä taulu koostuu noin 120 eri kentästä, joilla voidaan määritellä tuotteen perusominaisuuksia. Läheskään kaikki kentät eivät ole kuitenkaan pakollisia.
InventtableModule	Sisältää hintatiedot tuotteeseen liittyen. Tähän tauluun määritellään erikseen ryhminä myynti-, osto- ja varastohinta sekä niihin liittyvät mahdolliset alennukset ja muut tiedot. Vaikka hintoja ei vietäisi järjestelmään, täytyy tähän tauluun viedä merkintä jokaista tuotteen eri hintaryhmää varten nolla-arvoisena jotta tuotedata toimii oikein.
InventItemLocation	Sisältää logistisia tietoja tuotteista. Tänne tallennetaan esimerkiksi varastopaikka ja laskentaryhmät tuotteille. Jokaiselle tuotteelle pitää olla vähintään site- tasoinen määritys tässä taulussa jotta se toimii oikein tuotantotilauksilla.
InventItemPurchSetup	Sisältää perusvarastotiedot ostoon liittyen.

InventItemSalesSetup	Sisältää perusvarastotiedot myyntiin liittyen
InventItemInventSetup	Sisältää perusvarastotiedot varastointiin liittyen.

Käytännössä jokaisessa näistä tauluista on oltava tietoa tuotteeseen liittyen, jotta sitä voidaan oikeasti käyttää järjestelmässä. Nämä ovat myös ne taulut, joihin AX tekee automaattisesti merkinnän kun uusi tuote luodaan.

5.2 Reitti- ja rakennetaulujen määrittely

AX:ssä tuote voidaan määritellä joko item-, service- tai BOM-tyyppiseksi. Mikäli tuote on määritelty BOM-tyyppiseksi, se on niin kutsuttu rakennetuote. Rakennetuotteet koostuvat muista item-, service- tai BOM-tyyppisistä tuotteista. Yleensä niille määritellään myös tuotantoreitti, joka tarkoittaa sitä miten tuote käytännössä vaihe vaiheelta valmistetaan. BOM-tyyppiset tuotteet vaativat siis perustietotaulujen täyttämisen lisäksi erikseen oman rakenteen ja reittinsä. Lisäksi useita tuotteita voidaan liittää samaan rakenteeseen tai reittiin. Tämän takia reitit ja rakenteet on luotava omiin tauluihinsa.

Rakennetaulujen määrittäminen onnistuu helpoiten tutkimalla rakenne-näkymää. Se löytyy inventory management -moduulista kohdasta Bills of Materials. Tietolähteitä kyseiseltä näkymältä löytyy neljä kappaletta. Näistä tietolähteistä oleellisia ovat BOMTable ja BOMVersion -nimiset tietolähteet. Lisäksi näkymästä löytyy Lines-niminen painike. Tämän takaa aukeaa uusi näkymä, jossa näkyvät valittuun rakenteeseen liittyvät rakennerivit. Tämä näkymä on nimeltään BOMConsistOf ja sen tietolähteiden joukosta löytyy oleellinen tietolähde nimeltään BOM.

Taulukko 2. *Rakennetaulut.*

Taulun nimi	**Selite**
BOMTable	Sisältää rakenteet. Tähän tauluun tulee rakenteen nimi, yksilöivä id-tieto, tuoteryhmä, johon rakenne liittyy, sekä onko rakenne hyväksytty ja kuka rakenteen on hyväksynyt
BOMVersion	Sisältää tiedot rakenteeseen liittyvistä tuotteista. Tauluun tallennetaan sen BOM-tyyppisen tuotteen id-koodi, joka kyseiseen rakenteeseen halutaan liittää, sekä tieto siitä, onko versio hyväksytty ja kuka tuoteversion on hyväksynyt. Lisäksi tauluun voidaan tallentaa tiedot siitä, missä aikavälissä kyseinen tuoteversio on aktiivinen.
BOM	sisältää tiedot rakenteeseen liittyvistä rakennetuotteista. Tähän tauluun tallennetaan siis tiedot niistä tuotteista joita käytetään päätuotteen valmistamisessa. Tietoina tähän tauluun tulee esimerkiksi valmistuksessa kulutettu määrä, varasto josta tuotetta käytetään sekä tuotteen määräyksikkö.

Reittejä varten määriteltävät taulut ovat käytännössä melko samanlaiset kuin rakennetaulut. Reittitaulujen määrittäminen aloitetaan samaan tapaan kuin rakennetaulujenkin määrittäminen, eli avataan production-moduulista route details -näkymä. Tietolähteiden määrittely tapahtuu samalla tavalla kuin rakenteiden kanssa, reittien kohdalla reittivaiheet vain löytyvät route-painikkeen takaa. Itse reittivaiheet sisältävä route-näkymä eroaa kuitenkin jonkin verran vastaavasta rakennenäkymästä. Reittivaiheille voidaan vielä

lisäksi tallentaa omia vaiheversioita eri tuotteille. Tämä tarkoittaa käytännössä sitä, että esimerkiksi isommalle tuotteelle määritellään vaiheversiotauluun pidempi valmistusaika jne.

Taulukko 3. *Reittitaulut.*

Taulun nimi	Selite
RouteTable	Sisältää reitit. Tiedot ovat taulussa samaan tapaan kuin vastaavassa rakennetaulussa.
RouteVersion	Tähän tallennetaan tiedot reittiin liittyvistä tuotteista. Sisältää tiedot samaan tapaan kuin vastaava rakennetaulu.
Route	Tähän tallennetaan tiedot reittiin liittyvistä reittivaiheista. Reittivaiheet tallennetaan tauluun siihen järjestykseen kuin ne halutaan suorittaa, järjestys määritellään erillisellä rivinumerolla. Tähän tauluun tallennetaan lisäksi tietona esimerkiksi reittivaiheen operaatio sekä seuraava reittivaihe.
RouteOpr	Tähän tallennetaan eri reittivaiheiden erilaiset versiot. Käytännössä tauluun tallennetaan reitille linkitetyn tuotteen id-koodi siihen reittivaiheeseen, johon muutos halutaan tehdä. Tämän jälkeen tähän merkintään voidaan muokata reittivaiheen aikaa, valmistus määrää jne.

5.3 Excel-siirtopohjien luonti ja täyttäminen

Siirtopohjat tietoja varten on helpointa luoda käyttämällä siirtopohjavelhoa. Paras tapa on luoda vähintään kolme eri tyyppistä siirtopohjaan, yksi johon sijoitetaan tuotteen perustiedot sekä samanlaiset pohjat sekä reittejä että rakenteita varten. Pohjat kannattaa myös vastaisuuden varalta nimetä järkevällä tavalla, jotta tiedetään mitä ja milloin kyseisellä pohjalla on siirretty. Saman pohjan tietoja ei kannata korvata useaan kertaan. Mikäli tiedoissa myöhemmin havaitaan virhe, on kyseiset tiedot helppo ensin poistaa kannasta ja sen jälkeen palauttaa oikein nimetystä pohjasta.

Mikäli kyseinen siirtopohja halutaan luoda myöhemmin uudelleen, voidaan siitä tehdä velhon avulla määrittelyryhmä, jonka avulla voidaan definition groups -näkymästä ajaa ulos vastaavanlainen pohja. Siirtopohjaa luodessa voidaan myös halutessa poistaa ja lisätä valittujen taulujen kenttiä pohjaan. Mikäli tiedetään tarkasti mitä tietoja tauluun halutaan viedä, voidaan turhat kentät tässä vaiheessa poistaa. Tämä ei kuitenkaan ole välttämättä kannattavaa, sillä turhien kenttien poisto voidaan tehdä myös Excelissä yksinkertaisesti poistamalla turha sarake. Kenttien lisääminen Excelissä on huomattavasti vaativampi toimenpide. Jos jokin kenttä kuitenkin tarvitaan, on pohja käytännössä luotava uusiksi.

Pohjaan voidaan lisäksi siirtää tauluissa olemassa olevat tiedot niiden luonnin aikana. Näitä rivejä voidaan hyödyntää esimerkkiriveinä omia tietoja pohjaan siirrettäessä. Pohjaan siirrettävän tiedon määrää ei voida kuitenkaan muokata, joten mikäli tietoa kannassa on paljon, siirtyvät kaikki tiedot kannasta Exceliin ja pohjan luonti saattaa kestää pitkään. Tällöin kannattaa luoda esimerkiksi uusi tyhjä yritys ja tallentaa sinne yksi tai muutama tietorivi, jotka voidaan siirtää Excel-pohjaan. Tuotedatan kohdalla kannattaa luoda ainakin yksi item-tyyppinen tuote ja yksi BOM-tyyppinen

tuote. Näiden tuotteiden tiedot kannattaa myös täyttää mahdollisimman täydellisesti seuraavaa vaihetta varten.

Luoduissa pohjissa pakolliset kentät näkyvät selkeästi merkattuna erivärisinä kuin vapaavalintaiset kentät. Vähintään nämä kentät pitää täyttää, jotta tieto siirtyy kantaan kunnolla. Pohjassa on kuitenkin myös useita muita kenttiä, joiden täyttäminen on oleellista. Nämä kentät löytyvät helpoiten tarkastelemalla tietoja, jotka ovat siirtyneet ulos järjestelmästä. Joissain tauluissa saattaa myös olla samalle tuotteelle useampi merkintä, jolloin tieto siirtyy samasta taulusta useampaan eri tietokenttään esimerkiksi näkymillä. Näin on esimerkiksi inventtablemodule-taulussa, josta löytyy samalle id-koodille kolme eri merkintää, kaksi myynti- ja ostohintaa varten ja yksi varastohintaa varten. Pohjia täyttäessä on myös tärkeää huomioida että Excel-pohjaan siirretty tieto on samassa muodossa kuin kannassa. Esimerkiksi id-koodit täytyy ajaa kantaa string-muotoisina, eli Excel-pohjassa kyseisen kentän tulee olla tekstimuodossa. Mikäli kenttä on esimerkiksi numeromuodossa, ei tauluun siirry ollenkaan id-koodeja ja tiedot joudutaan siirtämään uudestaan.

5.4 Tiedon siirto ja validointi

Tiedon siirto on helpointa tehdä käyttämällä Excel import -näkymää. Näkymän kentistä oleellisia ovat file name, johon valitaan siirrettävä Excel-tiedosto sekä import rule, johon uusia tietoja siirrettäessä kannattaa valita vain insert only new records. Määrittelyryhmää ei tarvitse antaa.

Siirto kannattaa aloittaa perustietotauluista. Koska tauluihin ajetaan doInsert komento, ei periaatteessa ole merkitystä missä järjestyksessä tiedot ajetaan eri tauluihin. Päätaulu voidaan siis ajaa vaikka viimeisenä. Kuitenkin varmuuden vuoksi on viisasta ajaa taulut järjestelmään niin, että inventtable ajetaan ensimmäisenä, koska kaikista muista tauluista on relaatio tähän tauluun. Koska tauluja on paljon, voidaan ne halutessa siirtää kantaan esimerkiksi muutama kerrallaan. Tämä onnistuu muokkaamalla #Def-sivua Excel-pohjassa. Poistamalla taulun nimen tältä sivulta kyseinen taulu ei siirry järjestelmään. Myös siirtojärjestystä voidaan muokata tältä sivulta. Taulut siirtyvät järjestyksessä ylhäältä alaspäin.

Reitti- ja rakennetauluja siirrettäessä siirtojärjestyksellä ei myöskään ole suurta merkitystä. Mikäli reiteissä tai rakenteissa on jotain virheitä, on kuitenkin järkevää aloittaa siirto pelkästään itse reitti- ja rakennetauluista ja jättää molempien versiotaulu pois. Tällöin tiedon poisto virhetilanteen yhteydessä on helpompaa. Mikäli reittiin tai rakenteeseen on liitetty tuote tuoteversiotauluun, ei kyseistä reittiä tai rakennetta voi poistaa ennen kuin tuoteversio on poistettu. Jos kyseessä on iso siirto, saattaa tällainen poisto-operaatio vaatia paljon työtä. Myös reittien ja rakenteiden hyväksyntätiedot kannattaa asettaa jo Excelissä, sillä ohjelma ei anna hyväksyä kuin yhden reitin tai rakenteen ja niiden version kerralla.

Tietojen siirron jälkeen validointi eli tiedon oikeellisuuden tarkistaminen kannattaa aloittaa item details -näkymän kautta. Mikäli siirto on onnistunut, eli tauluun on ajettu tuotteet, mutta tiedoissa on ollut virheitä, tuotteet tai virheellinen tuote ei luultavasti näy tässä näkymässä. Vaikka tiedot näkyisivät item details -näkymässä, on hyvä vielä yrittää käyttää tuotetta eri tavalla ohjelmassa. Tämä tarkoittaa siis käytännössä esimerkiksi myynti- ja ostotilausten sekä tuotantotilausten luomista ja loppuun viemistä. Myös reitti- ja rakennetaulujen tarkastaminen onnistuu helpoiten käymällä läpi kyseisiin tauluihin liittyvät näkymät.

6 TILAAJIEN JA TOIMITTAJIEN IMPLEMENTOINTI MICROSOFT AX -JÄRJESTELMÄÄN

Tilaajat ja toimittajat ovat tuotetiedon ohella tuotannonohjausjärjestelmän oleellisimpia tietoja. Tietojen vienti AX:ään vaatii kuitenkin jonkin verran enemmän työtä kuin tuotedatan vienti, koska relaatiot eivät synny yhtä helposti kuin tuotetiedon kohdalla. Tällöin relaatiot täytyy luoda hyödyntämällä X++-ohjelmointikieltä. Tähän tarvittavat ohjelmakoodit ja niiden selitykset löytyvät tämän kappaleen lopusta.

6.1 Taulujen määrittely

Tilaaja- ja toimittajataulujen määrittely aloitetaan samaan tapaan kuin tuotetaulujenkin määrittäminen eli avaamalla näihin liittyvä näkymä. Oikea näkymä löytyy CRM-moduulista business relations -nimellä. Tähän näkymään kannattaa ensin yrittää luoda uusi tilaaja tai toimittaja, jolloin näkee helpommin vaadittavat kentät. Itse näkymä sisältää useita eri alisivuja sekä liittymiä muihin tauluihin.

AOT:sta kyseinen näkymä löytyy nimellä SmmBusRelTable. Tämä näkymä sisältää yhteensä 25 eri tietolähdettä, joten oikeiden tietolähteiden määrittäminen on tässä tilanteessa melko vaikeaa. Tällaisessa tilanteessa käytännössä ainoa tapa taulujen määrittelemiseen on ensin tarkistaa properties-ikkunan takaa jokaisen tietolähteen taulun nimi. Tämän jälkeen tarkistetaan, onko kyseisissä tauluissa tietoa. Helpoiten tämän tekeminen onnistuu ympäristössä, jossa ei vielä ole muuta tietoa. Tällöin voidaan järjestelmään syöttää yksi esimerkkitilaaja tai -toimittaja ja tarkistaa mihin tauluihin järjestelmä on tehnyt merkintöjä. Tässä tapauksessa suositeltavia täytettäviä tauluja on kolme. Lisäksi myöhemmin tullaan tarvitsemaan kolmea muuta lisätaulua.

Taulukko 4. O*soitetaulut.*

Taulun nimi	**Selite**
Address	Sisältää osoitteet eri henkilöille ja organisaatioille. Samalle henkilölle tai organisaatiolle voi olla useita eri osoitteita. Tällöin on hyvä määrittää pääasiallinen osoite.
DirPartyTable	Sisältää globaalin osoitekirjan. Tämä on siis osoitteiden päätaulu, joka yhdistelee eri taulujen tiedot yhteen, global addressbook- näkymään.
DirECommunicationAddress	Sisältää henkilöiden ja organisaatioiden yhteystiedot. Tämä tarkoittaa puhelinnumeroita, sähköpostiosoitteita jne.

DirPartyAddressRelationship	Sisältää relaation dirpartytable-tauluun. Tämän taulun avulla luodaan relaatio address-tauluun niin että address-taulussa voi olla useampia merkintöjä. Tähän tauluun ei viedä siirtovaiheessa tietoa, vaan sitä tarvitaan myöhemmin relaatioiden luomiseen.
DirPartyAddressRelationshipMap-ping	Sisältää dirpartyaddressrelationship- ja address- taulujen välisen relaation. Tähän tauluun ei viedä siirtovaiheessa tietoa, vaan sitä tarvitaan myöhemmin relaatioiden luomiseen.
DirPartyECommunicationRelationsh ip	Sisältää direcommunicationaddress -ja dirpartytable-taulujen välisen relaation. Tähän tauluun ei viedä siirtovaiheessa tietoa, vaan sitä tarvitaan myöhemmin relaatioiden luomiseen.

Tässä vaiheessa on hyvä huomata, että tilaajille ja toimittajille voidaan lisätä paljon muitakin tietoja, joita ei ole näissä tauluissa. Tällaisen oleellisia tietoja ovat esimerkiksi organisaationumero, VAT-numero, segmentit jne. Nämä tiedot löytyvät muista DirPartyTableen liittyvistä tauluista. Näitä tauluja ei kuitenkaan käsitellä tässä vaiheessa. Näihin tauluihin voidaan tehdä siirrot normaalisti sen jälkeen, kun kolmen perustaulun tiedot on saatu kuntoon.

Ennen Excel-siirtopohjien luontia täytyy näihin kolmeen tauluun luoda lisäkenttä. Tätä lisäkenttää tullaan tarvitsemaan seuraavassa vaiheessa, kun siirtopohjia täytetään. Lisäkenttään tulee laittaa yksilöivä tunnus jokaista tilaaja tai toimittajaa kohden. Tällä tunnistekentällä luodaan tietojen siirron jälkeen puuttuva relaatio näiden kolmen taulun välillä. Kenttä voidaan nimetä haluamalla tavalla ja kun siirto on kokonaan suoritettu, se voidaan poistaa turhana. Esimerkissä kenttä on nimetty accountID-nimellä.

6.2 Excel-siirtopohjien luonti ja täyttäminen

Siirtopohjat luodaan ja täytetään samaan tapaan kuin tuotetietotaulutkin. Tässä vaiheessa huomataan, että erityisesti yhteystietotaulu ja osoitetaulu sisältävät melko paljon samoja kenttiä. Suositeltavaa on, että molempiin tauluihin täytetään kuitenkin tiedot, jotta AX osaa varmasti näyttää kaikki tiedot oikein jokaisessa näkymässä. Kenttiä täytettäessä on hyvä olla erityisen tarkkana siitä mihin kenttään laitetaan pelkästään numeroita ja mihin sekä numeroita että kirjaimia. Suurin osa kentistä hyväksyy luonnollisesti molemmat, mutta tietotyyppien kanssa kannattaa kuitenkin olla tarkkana.

Tässä vaiheessa tarvitaan myös jokaiselle tilaajalle ja toimittajalle aikaisemmin luotua tunnistekenttää varten yksilöivä tunniste. Periaatteessa tunnisteena voidaan käyttää esimerkiksi juoksevaa numerointia. Tällöin numerot täytetään Excel-pohjaan. Täyttäminen aloitetaan Dirpartytable-taulusta. Näin saatu yksilöivä tunniste lisätään sen jälkeen osoite- ja yhteystietotauluihin niihin liittyviin tietoihin. Mikäli osoite- tai yhteystietoja on useampia yhdelle tilaajalle tai toimittajalle, lisätään tunniste silloin yksinkertaisesti useampaan kertaan näihin tauluihin.

6.3 Tiedon siirto ja validointi

Taulujen siirtojärjestyksellä ei ole tässäkään tapauksessa suurta merkitystä. Tietojen siirtäminen kannattaa kuitenkin aloittaa Dirpartytable-taulusta.

Myös muita tauluja kuin nämä kolme voidaan siirtää samassa siirrossa, mutta suositeltavaa on kuitenkin siirtää vain nämä ensin. Tällöin tiedon määrä pysyy vähäisenä ja virheet on helppo korjata.

Validointivaiheessa voidaan tarkistaa sekä business relations -näkymä, että global address book -näkymä. Näissä tulisi näkyä jokainen dirpartytableen lisätty tilaaja tai toimittaja. Tässä vaiheessa kuitenkin addresses- ja contact info -välilehdet ovat vielä tyhjiä. Tiedot ovat oikeasti olemassa, mutta relaatio niiden väliltä puuttuu.

Tämä relaatio luodaan dirpartyaddressrelationshipmapping-tauluun joka sisältää jokaista osoitetietoa kohti avainkentän. Relaatio luodaan dirpartyaddressrelationship-taulun kautta. Tämä taulu taas kokoaa useammat eri merkinnät saman asiakkaan taakse. Tällä avainkentällä on toinen avainpari josta löytyy dirpartytable-taulussa oleva tieto. Koska avainkenttänä toimii Recid-niminen systeemikenttä, tätä relaatioita ei voida luoda siirron yhteydessä. Yhteystiedot toimivat samalla tavalla, mutta avainkenttä löytyy direcommunicationrelationship-taulusta ja välissä ei ole kolmatta taulua.

6.4 Puuttuvien relaatioiden luonti

Puuttuvat relaatiot luodaan liitteinä löytyvien X++-koodien avulla. Koodit voidaan suorittaa AOT:n kautta hyödyntämällä sieltä löytyvää jobs-solmua. Jobit ovat käytännössä pieniä koodinpätkiä, jotka suoritetaan ilman luokkarakenteita. Ne soveltuvat tällaisiin tehtäviin hyvin. Jobien suorituksessa kannattaa olla tarkkana, sillä niiden tekemiä muutoksia kantaan ei luonnollisesti voida peruuttaa. Tämän lisäksi kantamuutokset saattavat ohittaa joitakin kenttien eheyssääntöjä, joten muokatun tiedon tulee olla tarkasti oikeassa muodossa.

Liitteessä 1 oleva koodi on nimeltään DirPartyRelations. Sillä luodaan yhteys address- ja dirpartytable-taulujen välille. Koodin alussa luodaan tarvittavat muuttujat jokaiselle neljälle vaadittavalle taululle. Itse koodi valitsee jokaisen address-taulun rivin ja hakee tälle vastaavaa tietoa dirpartytable-taulusta accountid-kentän perusteella. Mikäli vastaavuus löydetään, luodaan näiden tietueiden välille relaatio dirpartyaddressrelationshipmapping- ja dirpartyaddressrelationship-taulujen kautta täyttämällä vaaditut kentät. Liitteessä 2 olevassa DirCommRelations-metodilla luodaan taas relaatio direcommunicationdetails- ja dirpartytablen-taulujen välille. Tämä relaatio luodaan DirPartyECommunicationRelationship-tauluun.

7 YHTEENVETO

Tiedon implementointi projektin aikana osoittautui huomattavasti haastavammaksi tehtäväksi kuin olin aluksi kuvitellutkaan. Minulla ei ollut käytännössä ollenkaan kokemusta ERP-järjestelmistä eikä Dynamics

AX:stä, joten hyvin suuri osa projektiin käytetystä ajasta meni pelkästään uusien asioiden sisäistämisessä. Käytin myös paljon täysin uusia ohjelmistoja sekä ohjelmointiympäristöjä. Projektin alussa olinkin jonkin verran epävarma sen onnistumisesta ilman ulkopuolista tukea. Olen kuitenkin lopulta tyytyväinen siihen, että sain itse suhteellisen rauhallisella aikataululla toteuttaa omaa osa-aluetta projektistani, sillä se oli hyvin opettavainen kokemus.

Kuitenkin kaikista suurin ongelma koko työn aikana oli puuttuvat ohjeistukset. Tiedon implementoinnista AX:ään on vain hyvin vähän dokumentaatiota. Dokumentit käsittelevät pääasiassa vain tiedonsiirtotyökalun käyttöä perustasolla. Tämä tarkoittaa sitä, että esimerkiksi tuotedatan siirrosta ei erikseen löytynyt minkäänlaisia ohjeita ja suurin osa selvitystyöstä oli yksinkertaisesti tehtävä hyvin vähäisellä avustuksella. Käytännössä ohjelman IDE:n opettelu vähintään perustasolla vaaditaan, jotta tiedonsiirto voidaan toteuttaa.

Käytännön työn aikana sain onneksi hyvin vapaat kädet siirron toteuttamiseen. Testiympäristöt toimivat hyvin, joten tiedonsiirron testaaminen oli helppo toteuttaa. Tiedonsiirtoprojektissa testiympäristö onkin miltei välttämätön, sillä järjestelmän tiedot saattavat korruptoitua täysin mikäli siirto epäonnistuu. Siirtovaihe olikin toteutettu kokonaisuudessaan testiympäristöön useita kertoja ennen varsinaiseen tuotantoympäristöön siirtämistä. Lopulta itse tuotantoympäristöön siirto onnistui erittäin hyvin, virheitä sattui vähän ja kaikki virheet olivat paikattavissa pienillä korjaussiirroilla ilman koko kannan tyhjentämistä. Tietojen siirtäminen onnistui myös pääosin automaattisesti eli järjestelmässä oli vain joitakin tietoja, joita täytyi korjata käsin.

Opinnäytetyön kirjoittamista aloittaessani oli käytännön osuus jo saatu tehtyä eli olin jo toteuttanut tiedonsiirron GS-Hydron AX-tuotantoympäristöön. Suurin etu itse opinnäytetyön tekemisestä oli omien tietojeni syventyminen. En vielä työtä aloittaessa tuntenut niin hyvin kaikkien taustalla toimivien järjestelmien toimintaa, vaikka osasinkin toteuttaa tiedonsiirron AX:ään. Lisäksi koin itse hyvin hyödyllisenä toteuttaa tällaisen ohjeistuksen, koska ohjeistuksen löytäminen tähän tiedonsiirtoon AX-ympäristöön tuntui olevan mahdotonta. Tämä toi lisää motivaatiota työn kirjoittamiseen.

LÄHTEET

Andreasen, Steen. 2006. MORPHX IT an introduction to Axapta X++ and the MorphX Development Suite.

Cognos Oy. 2002. Cognos Series7 PowerPlay for User – Käyttäjän ohje.

GS-Hydro Oy. 2009. http://www.gshydro.com/

Garcia-Molina, Hector, Ullman, Jeffrey D, & Widom, Jennifer. 2009. Database systems: the complete book. Upper Saddle River, N.J, Pearson Prentice Hall , cop.

Hamilton, Scott. 2009. Managing your supply chain using Microsoft Dynamics.

Kline, Kevin E., Gould, Lee & Zanevsky Andrew. 1999. Transact-SQL programming. O'Reilly Media, Inc.

Melton, Jim & Simon, Alan R. 2002. SQL:1999:
Understanding relational language components. Morgan Kaufmann.

Microsoft Corporation. 2009. Microsoft Official Training Materials for Microsoft Dynamics ®.

The Microsoft Dynamics AX Team. 2009. Inside Microsoft Dynamics 2009. Microsoft Corporation.

Microsoft Dynamics AX 4 Tables, Maps and Views. 2010. Viitattu 28.4.2010.http://msdn.microsoft.com/en-US/library/bb395036(v=AX.10).aspx

Monk, Ellen F. & Wagner, Bret J. 2009. Concepts in enterprise resource planning. Boston : Course Technology Cengage Learning, cop.

SQL Server Books Online. 2010. Viitattu 5.5.2010. http://msdn.microsoft.com/en-us/library/ms130214(v=sql.105).aspx

LIITE 1

```
static void DirPartyRelations(Args _args)
{

Address address;
DirPartyTable dirpartytable;
DirPartyAddressRelationship dirpartyaddressrel;
DirPartyAddressRelationshipMapping dirpartyaddressrelmap;
;
ttsbegin;
while select forupdate address
{
  select forupdate dirpartytable where address.WMD_cust == dirpartyta
  ble.WMD_cust;
  dirpartyaddressrel.partyid = dirpartytable.PartyId;
  dirpartyaddressrel.IsPrimary = address.WMD_Primary;
  dirpartyaddressrel.Status = DirPartyAddressRelationshipStatus::Active;
  dirpartyaddressrel.ValidToDateTime = DateTimeUtil::maxValue();
  dirpartyaddressrel.ValidFromDateTime = Datetimeutil::minValue();
  dirpartyaddressrel.Shared = NoYes::Yes;
  dirpartyaddressrel.insert();
  address.AddrRecId = dirpartytable.RecId;
  dirpartyaddressrelmap.AddressRecId = address.RecId;
  dirpartyaddressrelmap.RefCompanyId = address.dataAreaId;
  dirpartyaddressrelmap.PartyAddressRelationshipRecId = dirpartyaddress-
  rel.RecId;
  dirpartyaddressrelmap.insert();
  address.update(false,false);
}
ttscommit;

}
```

```
static void DirCommRelations(Args _args)
{

DirPartyECommunicationRelationship DirPECR;
DirECommunicationAddress DirECA;
DirPartyTable DirPT;
;
ttsbegin;
while select forupdate DirECA
{
   select DirPT where DirPT.WMD_cust == DirECA.WMD_cust;
   DirPECR.PartyId = DirPT.PartyId;
   DirPECR.Status = DirECommunicationStatus::Active;
   DirPECR.ValidFromDateTime = Datetimeutil::minValue();
   DirPECR.ValidToDateTime = Datetimeutil::maxValue();
   DirPECR.Priority = DirECommunicationPriority::Low;
   DirPECR.ValuesRecId = DirECA.RecId;
   DirPECR.PrivacyGroupId = "Public";
   DirPECR.insert();
}
ttscommit;

}
```